NAME:KRANTI BHOSALE ROLL NO:I3122
PROJECT DATABASE:POKEMON

AIM:. Develop a mini project in a group using different predictive models techniques to solve any real life problem. (Refer link dataset- https://www.kaggle.com/tanmoyie/us-graduate-schools- admissionparameters)

INFORMATION ABOUT MY PROJECT AND DATABSE:This data set includes 721 Pokemon, including their number , name, first and second type, and basic stats: HP, Attack, Defense, Special Attack, Special Defense, and Speed. It has been of great use when teaching statistics to kids. With certain types you can also give a geeky introduction to machine learning.

#: ID for each pokemon
Name: Name of each pokemon
Type 1: Each pokemon has a type, this determines weakness/resistance to attacks
Type 2: Some pokemon are dual type and have 2
Total: sum of all stats that come after this, a general guide to how strong a pokemon is
HP: hit points, or health, defines how much damage a pokemon can withstand before fainting
Attack: the base modifier for normal attacks (eg. Scratch, Punch)
Defense: the base damage resistance against normal attacks
SP Atk: special attack, the base modifier for special attacks (e.g. fire blast, bubble beam)
SP Def: the base damage resistance against special attacks
Speed: determines which pokemon attacks first each round

In [1]:

```
import pandas as pd
import numpy as np
df = pd.read_csv("Pokemon.csv")
df
```

Out[1]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Ge
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	
...	...	...	...	...	...	...	...	...	...	...	...	
795	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	
796	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	
799	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	

800 rows × 13 columns

In [2]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   #           800 non-null    int64 
 1   Name        800 non-null    object
 2   Type 1      800 non-null    object
 3   Type 2      414 non-null    object
 4   Total       800 non-null    int64 
 5   HP          800 non-null    int64 
 6   Attack      800 non-null    int64 
 7   Defense     800 non-null    int64 
 8   Sp. Atk     800 non-null    int64 
 9   Sp. Def     800 non-null    int64 
 10  Speed       800 non-null    int64 
 11  Generation  800 non-null    int64 
 12  Legendary   800 non-null    bool  
dtypes: bool(1), int64(9), object(3)
memory usage: 75.9+ KB
```

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
#               0
Name            0
Type 1          0
Type 2        386
Total           0
HP              0
Attack          0
Defense         0
Sp. Atk         0
Sp. Def         0
Speed           0
Generation      0
Legendary       0
dtype: int64
```

In [4]:
```
df.head(6)
```

Out[4]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generati
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	
5	5	Charmeleon	Fire	NaN	405	58	64	58	80	65	80	

In [5]:
```
df.columns
```

Out[5]:
```
Index(['#', 'Name', 'Type 1', 'Type 2', 'Total', 'HP', 'Attack', 'Defe
nse',
       'Sp. Atk', 'Sp. Def', 'Speed', 'Generation', 'Legendary'],
      dtype='object')
```

In [6]:
```
data2=df.copy()
```

In [7]:
```
data2=data2.fillna(data2.mean())      #replace null values with mean
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_7396\2591363272.py:1: Futu
reWarning: The default value of numeric_only in DataFrame.mean is depr
ecated. In a future version, it will default to False. In addition, sp
ecifying 'numeric_only=None' is deprecated. Select only valid columns
or specify the value of numeric_only to silence this warning.
  data2=data2.fillna(data2.mean())      #replace null values with mean
```

In [8]:
```
data2.head()
```

Out[8]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generati
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	

In [9]:
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   #           800 non-null    int64
 1   Name        800 non-null    object
 2   Type 1      800 non-null    object
 3   Type 2      414 non-null    object
 4   Total       800 non-null    int64
 5   HP          800 non-null    int64
 6   Attack      800 non-null    int64
 7   Defense     800 non-null    int64
 8   Sp. Atk     800 non-null    int64
 9   Sp. Def     800 non-null    int64
 10  Speed       800 non-null    int64
 11  Generation  800 non-null    int64
 12  Legendary   800 non-null    bool
dtypes: bool(1), int64(9), object(3)
memory usage: 75.9+ KB
```

In [10]:
```
dist=(data2['Name'])
distset=set(dist)
dd=list(distset)
dictOfWords = { dd[i] : i for i in range(0, len(dd) )}
data2['Name']=data2['Name'].map(dictOfWords)
```

In [11]:
```
dist=(data2['Type 1'])
distset=set(dist)
dd=list(distset)
dictOfWords = { dd[i] : i for i in range(0, len(dd) )}
data2['Type 1']=data2['Type 1'].map(dictOfWords)
```

In [12]:
```
data2["Type 1"]=data2["Type 1"].fillna(data2["Type 1"].mean())
```

In [13]:
```
data2
```

Out[13]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	L
0	1	532	9	Poison	318	45	49	49	65	65	45	1	
1	2	53	9	Poison	405	60	62	63	80	80	60	1	
2	3	31	9	Poison	525	80	82	83	100	100	80	1	
3	3	426	9	Poison	625	80	100	123	122	120	80	1	
4	4	123	14	NaN	309	39	52	43	60	50	65	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	
795	719	138	13	Fairy	600	50	100	150	100	150	50	6	
796	719	720	13	Fairy	700	50	160	110	160	110	110	6	
797	720	751	3	Ghost	600	80	110	60	150	130	70	6	
798	720	452	3	Dark	680	80	160	60	170	130	80	6	
799	721	23	14	Water	600	80	110	120	130	90	70	6	

800 rows × 13 columns

In [14]:
```
data2.isnull().sum()
```

Out[14]:
```
#               0
Name            0
Type 1          0
Type 2        386
Total           0
HP              0
Attack          0
Defense         0
Sp. Atk         0
Sp. Def         0
Speed           0
Generation      0
Legendary       0
dtype: int64
```

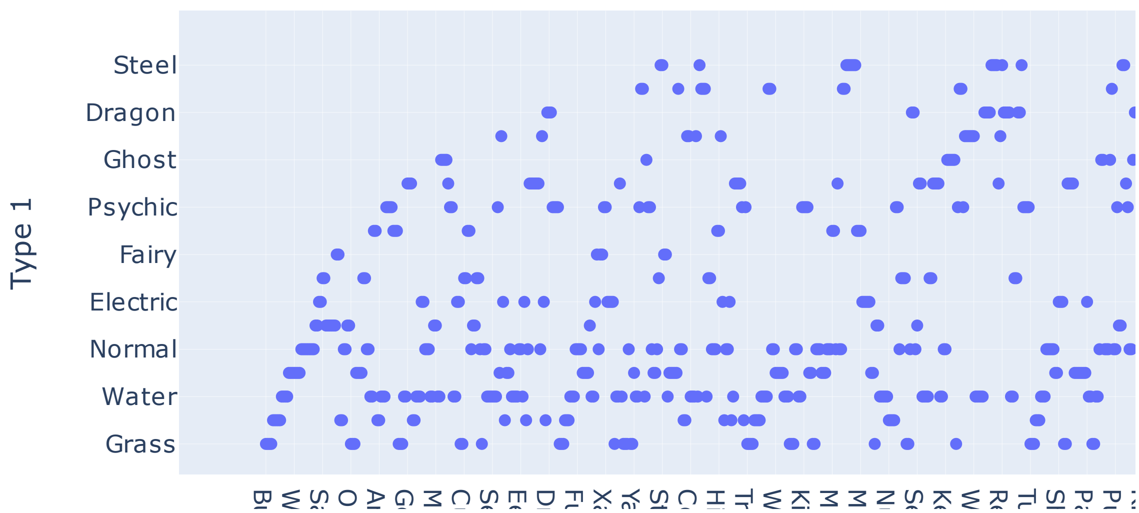
In [15]:
```
data2 = data2.drop('Legendary' , 1)
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_7396\1980597150.py:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  data2 = data2.drop('Legendary' , 1)
```

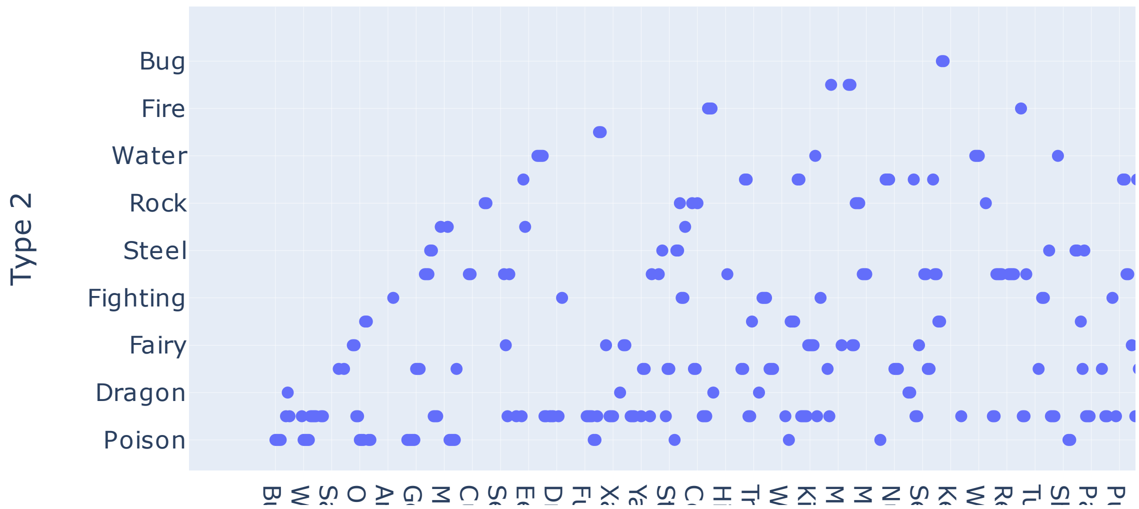
In [16]:
```
data2.columns
```

Out[16]:
```
Index(['#', 'Name', 'Type 1', 'Type 2', 'Total', 'HP', 'Attack', 'Defe
nse',
       'Sp. Atk', 'Sp. Def', 'Speed', 'Generation'],
      dtype='object')
```

In [17]:
```
import plotly.express as px
#EDA (Analyse the data)
fig = px.scatter(df, x="Name", y="Type 1")     #Plotting the Bubble Chart
fig.show()
```



In [23]:
```
import plotly.express as px
#EDA (Analyse the data)
fig2 = px.scatter(df,y="Type 2",x="Name")     #Plotting the Bubble Chart
fig2.show()
```

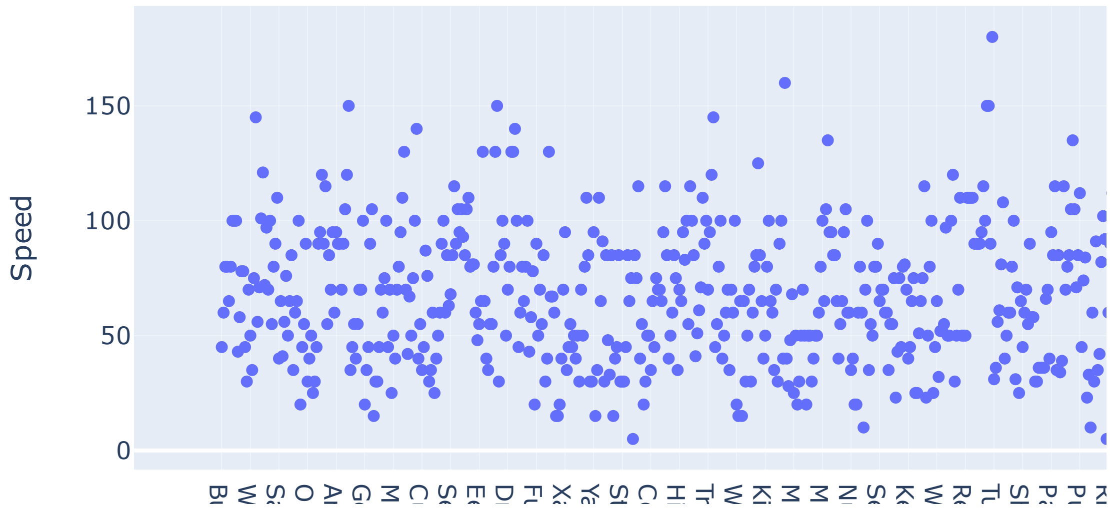


In [18]:
```
data2.columns
```

Out[18]:
```
Index(['#', 'Name', 'Type 1', 'Type 2', 'Total', 'HP', 'Attack', 'Defe
nse',
       'Sp. Atk', 'Sp. Def', 'Speed', 'Generation'],
      dtype='object')
```

In [19]:
```
features = data2[[ 'Type 1', 'Total', 'HP', 'Attack', 'Defense',
       'Sp. Atk', 'Sp. Def', 'Speed']]
labels = data2['Name']
```

In [21]:
```
import plotly.express as px
#EDA (Analyse the data)
fig2 = px.scatter(df,x="Name",y="Speed")    #Plotting the Bubble Chart
fig2.show()
```



In [25]:
```
#splitting into train & test data

from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest = train_test_split(features,labels,test_siz
```

In [26]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn import tree
```

In [27]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(Xtrain,Ytrain)
print(regr.predict(Xtest))
```

```
[377.28773691 487.30563162 407.44556805 365.05121072 495.37462501
 363.14786394 375.17259164 357.70598351 360.79854622 400.91753312
 362.50236699 368.8091812  425.01126568 391.63292631 382.52549329
 382.58488195 401.88762052 375.73535784 391.19762927 388.07614691
 363.58698041 501.31506236 390.0742446  440.76675168 390.08496562
 358.66645372 362.02837919 390.37078931 360.393137   465.82939905
 426.48757379 392.44522574 387.96620404 386.31145859 379.40669653
 373.34553847 397.63493528 433.88239215 364.13112542 476.80480079
 439.98263121 407.54473699 459.42312688 377.84153374 417.38643291
 400.52147704 420.66768628 410.18545548 428.21214485 372.2661843
 378.94037359 466.2127055  401.43777872 384.2960865  360.31424782
 366.28692183 361.35711633 358.32616136 382.58085261 375.31602641
 391.52955295 403.347513   422.05799333 379.96879821 397.03394739
 403.06701543 387.61819601 434.22177766 360.01385386 360.79395835
 388.07614691 379.78171066 387.55670167 424.54752397 377.42600087
 398.40498167 413.87822636 398.35155401 384.40622915 416.36632329
 385.634831   411.04466379 378.96825437 374.56945174 360.96780521
 432.32246639 388.40896835 390.55094808 400.28835711 387.6980764
 361.78180771 396.94247591 379.29546486 498.51681979 361.64982045
 384.82907939 387.35395705 372.17609946 358.65393474 384.6661446
 386.25918941 375.07549737 406.80621073 436.28612495 424.18946027
 377.1068814  399.4900013  393.9718089  479.02218219 451.23218996
 389.75801041 380.05083821 378.69449943 419.37639188 473.58980408
 356.9083504  375.32683724 426.21635843 403.92209364 374.58190558
 464.33448786 360.55244784 389.02189178 367.70527899 428.80671533
 393.77133517 396.57473754 360.3725385  381.12387746 382.54923607
 452.56631323 400.3712703  390.80087054 358.53188802 391.06923692
 382.82770409 360.79854622 416.02335629 431.75912909 397.55353884
 391.36554292 356.58378032 364.53497917 375.08867301 392.91330104
 387.6649443  458.87991049 496.9924109  371.12939799 459.00842482
 395.48999259 360.92559808 411.07459609 397.27569828 369.1883138
 375.30285233 376.62626376 387.59706892 374.8038133  370.52229598]
```

In [28]:
```
y_pred=regr.predict(Xtest)
```

In [29]:
```
from sklearn.metrics import r2_score
r2_score(Ytest, y_pred)
```

Out[29]:
```
0.012250271259021672
```

In [ ]: